

日 本 国 特 許 庁
JAPAN PATENT OFFICE



別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出 願 年 月 日

Date of Application:

2001年 9月25日

出 願 番 号

Application Number:

特願2001-290986

出 願 人

Applicant(s):

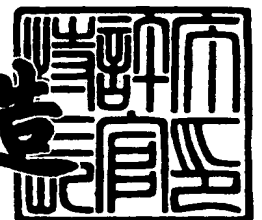
株式会社日立製作所

U.S. Appln. Filed 2-15-02
Inventor: S. Misaka et al
Mattingly Stanger & Malur
Docket NIT-321

2001年12月 7日

特 許 庁 長 官
Commissioner,
Japan Patent Office

及 川 耕 造



出証番号 出証特2001-3107554

【書類名】 特許願
 【整理番号】 NT01P0324
 【提出日】 平成13年 9月25日
 【あて先】 特許庁長官 殿
 【国際特許分類】 G06F 9/30
 G06F 12/02
 G06F 9/42

【発明者】

【住所又は居所】 東京都国分寺市東恋ヶ窪一丁目280番地 株式会社日立製作所 中央研究所内

【氏名】 三坂 智

【発明者】

【住所又は居所】 東京都国分寺市東恋ヶ窪一丁目280番地 株式会社日立製作所 中央研究所内

【氏名】 相坂 一夫

【特許出願人】

【識別番号】 000005108

【氏名又は名称】 株式会社日立製作所

【代理人】

【識別番号】 100068504

【弁理士】

【氏名又は名称】 小川 勝男

【電話番号】 03-3661-0071

【選任した代理人】

【識別番号】 100086656

【弁理士】

【氏名又は名称】 田中 恭助

【電話番号】 03-3661-0071

【選任した代理人】

【識別番号】 100094352

【弁理士】

【氏名又は名称】 佐々木 孝

【電話番号】 03-3661-0071

【手数料の表示】

【予納台帳番号】 081423

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【ブルーフの要否】 要

【書類名】 明細書

【発明の名称】 異種実行環境における計算機システム構成方法、および、異種実行環境で実行させるためのプログラム

【特許請求の範囲】

【請求項 1】 計算機システムの異なった実行環境で実行させるときのアプリケーション・プログラムにエラーコードを返す際の異種実行環境における計算機システム構成方法において、

この計算機システムは、ハードウェアとして汎用レジスタを有する CPU と、そのハードウェア上で実行させるオペレーティングシステムとを持ち、

前記 CPU の命令セットとしては、前記汎用レジスタに Immediate 値を格納する Immediate ロード命令を含み、

前記オペレーティングシステム上で実行させる実行プログラムとしては、前記アプリケーション・プログラムに対応する実行プログラムと、異なった実行環境の差異を吸収する実行環境差異吸収プログラムとからなり、

前記実行環境差異吸収プログラムから前記アプリケーション・プログラムに対応する実行プログラムに、制御が戻るときに返すエラーコードを実行環境によらない共通化エラーコードとして定義しておき、

しかも、この共通化エラーコードを、前記 Immediate ロード命令で設定可能な数値範囲内の値として定め、

前記共通化エラーコードを、前記 Immediate ロード命令の命令コード内に保持することを特徴とする異種実行環境における計算機システム構成方法。

【請求項 2】 前記共通化エラーコードを、前記 Immediate 値の MSB をゼロとした数値範囲内で定めることを特徴とする請求項 1 記載の異種実行環境における計算機システム構成方法。

【請求項 3】 前記命令セットの仕様として、前記汎用レジスタのビット数より小さいデータを、前記汎用レジスタにロードするときに、そのデータの先頭部を自動的にゼロ拡張するようになっているときに、

前記共通化エラーコードを、符号無し of 正の数で定めることを特徴とする請求項 1 記載の異種実行環境における計算機システム構成方法。

【請求項4】 前記命令セットの仕様として、前記汎用レジスタのビット数より小さいデータを、前記汎用レジスタにロードするときに、そのデータの先頭部を自動的に符号拡張するようになっているときに、

前記共通化エラーコードを、前記 Immediate 値の MSB をゼロとした数値範囲内で定めることを特徴とする請求項1記載の異種実行環境における計算機システム構成方法。

【請求項5】 計算機システムの異なった実行環境で実行させるときのアプリケーション・プログラムにエラーコードを返す際の異種実行環境で実行させるためのプログラムにおいて、

この計算機システムは、ハードウェアとして汎用レジスタを有する CPU と、そのハードウェア上で実行させるオペレーティングシステムとを持ち、

前記 CPU の命令セットとしては、前記汎用レジスタに Immediate 値を格納する Immediate ロード命令を含み、

前記オペレーティングシステム上で実行させる実行プログラムとしては、前記アプリケーション・プログラムに対応する実行プログラムと、異なった実行環境の差異を吸収する実行環境差異吸収プログラムとからなり、

前記実行環境差異吸収プログラムから前記アプリケーション・プログラムに対応する実行プログラムに、制御が戻るときに返すエラーコードを実行環境によらない共通化エラーコードとして定義しておき、

しかも、この共通化エラーコードを、前記 Immediate ロード命令で設定可能な数値範囲内の値として定め、

前記共通化エラーコードを、前記 Immediate ロード命令の命令コード内に保持して、

前記アプリケーション・プログラムに対応する実行プログラム内のコードと前記実行環境差異吸収プログラム内のコードにのみ共通化エラーコードの情報を含むことを特徴とする異種実行環境で実行させるためのプログラム。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

本発明は、異種実行環境における計算機システム構成方法に係り、アプリケーション・プログラムに、異なった実行環境の共通のエラーコードを返すようにするときに、実行プログラムのメモリ効率がよく、実行速度を低下させることなく、計算機システムを実装できるような計算機システム構成方法に関する。

【 0 0 0 2 】

【従来の技術】

従来、異なった環境でコンピュータシステムを実行させるために、開発や移植を容易におこなうための技術が提案されている。

【 0 0 0 3 】

このような技術の例として、特開平 8 - 6 3 3 6 3 号公報の「仮想実行環境システム」がある。

【 0 0 0 4 】

以下では、図 1 を用いてこの仮想実行環境システムについて説明する。

図 1 は、特開平 8 - 6 3 3 6 3 号公報の「仮想実行環境システム」のシステム概要図である。

【 0 0 0 5 】

ユーザ（プログラマ）が作成するアプリケーション・ソフトウェアのソースコードとしては、プログラム本体 1 1 と置換情報記述部 1 2 がある。

【 0 0 0 6 】

プログラム本体は、アプリケーション・ソフトウェアの目的ごとに異なるソースコードであり、置換情報記述部 1 2 は、プログラム本体 1 1 中に記述された情報を、仮想実行環境が実現される既存オペレーティング・システム 6 0 の機種毎に対応して適切な情報に置換える部分である。

【 0 0 0 7 】

オペレーティング・システム 6 0 で実行するためには、これらのプログラム本体 1 1 と置換情報記述部 1 2 に記述された置換情報を、既存オペレーティング・システム用コンパイラを利用して、実行可能プログラム 3 0 にし、それを実行部 2 2 が実行する。翻訳部 2 1 は、そのための仮想実行環境上で実行可能な実行可能プログラムに翻訳する部分である。

【0008】

その際に、ソースプログラムをC言語で記述するときには、置換情報記述部12は、コンパイラのプリプロセッサとして処理されるマクロ定義を利用することができる。

【0009】

このようにして、アプリケーションソフトウェア10のプログラム本体11を変更する事無く、様々な環境用に作成された実行可能プログラム30を既存オペレーティング・システム60上で動作させることが可能となる。

【0010】

次に、図2を用いて、より詳細に異種実行環境において従来技術の計算機システムを構成し、実行する場合について説明する。

図2は、異種実行環境においてソフトウェア開発をおこなうときの概念図である。

【0011】

ここでは、C言語により、アプリケーションを開発する場合を想定する。図1の置換情報記述部12にあたる部分に、C言語の共通関数102が提供される。

【0012】

システムとして、図2の様に計算機システム-A105a、計算機システム-B105bがあり、そこでプログラムを実行する環境を実行環境-A104a、実行環境-B104bと言うことにする。実行環境-A104aは、具体的には、ハードウェア-A106aと、ハードウェアを制御しアプリケーションプログラムとハードウェアの仲立ちをするOS-A107aである。

【0013】

アプリケーション本体100は、ユーザが個々のアプリケーションを記述するソースコードであり、システムから機能の提供を受けるときには、共通化関数102を呼出す。ここで、注意することは、アプリケーション本体100を記述するユーザは、共通化関数インタフェース101を意識すればよく、個々の環境によってコーディングを変えなくても良いことである。したがって、アプリケーション本体100は、ソースレベルでの互換性が保たれることになる。

【0014】

共通化関数102、開発用ライブラリ関数として提供され、ユーザには、共通化関数インタフェースが公開されることになる。共通化関数は、この様に環境に依らない統一したインタフェースをユーザに提供するためのものである。

【0015】

そして、ユーザが記述したソースをコンパイル・リンク作業をおこなって、実行形式を作成することになるが、実行環境-A104aで実行させるときには、図2に示される様に、コンパイラ-A108aでコンパイルし、実行環境-B104bで実行させるときには、コンパイラ-B108bでコンパイルする。

【0016】

また、同様に実行環境-A用のリンカA109aにより、実行環境プログラム-A103aをリンクする。実行環境プログラム-A103aは、実行環境-A104aで実行させるためのプログラムであり、このなかで、OS-A107aのシステムコールがコールされることになる。実行環境-B104bについても同様である。

【0017】

ユーザのソースコードであるアプリケーション本体100に対して、コンパイル、リンクがおこなわれ、この例では、実行環境-A104aと実行環境-B104b用に、それぞれ実行プログラム-A110aと、実行プログラム-B110bが生成される。

【0018】

そして、共通化関数102をコンパイルしたモジュールと、実行環境プログラム-A103a、実行環境プログラム-B103bは、コンパイル・リンクの作業を経て、それぞれの固有の実行環境で実行可能な実行環境差異吸収プログラム-A111aと実行環境差異吸収プログラム-B111bになる。

【0019】

次に、このようにして作成した実行プログラムを、計算機システムの実行環境のもとで動作させる場合について概説する。

【0020】

例えば、実行プログラム-A110aを、実行環境-Aのもとで実行するときには以下ようになる。すなわち、実行プログラム-A110aでは、共通化インターフェース102により、実行環境差異吸収プログラム-A111aを呼出す。この実行環境差異吸収プログラム-A111aは、アプリケーションから見たOSの差異を吸収し、インターフェースを共通化されるために設けられるモジュールである。呼び出された実行環境差異吸収プログラム-A111aは、OS-A107aの機能を実現するためのシステムコールを呼出して、所望のOSの機能をハードウェア-A106aにより実現する。ハードウェア-A106aの動作が終了すると、OS-A107aは、実行環境差異吸収プログラム-A111aに制御を戻す。最後に、実行環境差異吸収プログラム-A111aは、制御を実行プログラム-A110aに戻す。

【0021】

実行プログラム-B110bを、実行環境-B104bのもとで実行させるときも、同様の動作をするが、実行環境-B104bは、計算機システム-Bでさせるときの環境であるので、前提となるOSやハードウェアが異なっている。例えば、ハードウェア-A106aとハードウェア-B106bでは、機械語命令体系が異なり、実行環境プログラム-A103aと実行環境プログラム-B103bでは、OSの機能と呼出すためのシステムコールが異なっている。

【0022】

共通化インターフェース102は、プログラム本体11に変更を加える事無く、異なるOS上に実装してもプログラム本体11から、ほぼ同一のOS機能と呼び出して、異なるOS107に依存しない共通化エラーコードをプログラム本体11に伝えることを可能にするものである。なお、ここで、共通化エラーコードと言っているのは、OSに依存するエラーコードを、実行環境差異吸収プログラムで、そのまま、あるいは、書き換えて、アプリケーションから見たときには、環境によらない共通化されたエラーコードにするためである。

【0023】

共通化インターフェース102の仕様を定義することでOS機能の呼び出し方やプログラミング記述が統一され、プログラム本体11を別のアプリケーションソ

フトウェアに再利用させることが容易になる。

【0024】

【発明が解決しようとする課題】

上記従来技術は、異なった計算機システムの環境に、アプリケーションソフトウェアを開発して実行するシステムに関するものであった。

【0025】

ここで、図3ないし図7を用いて上記従来技術の動作において、エラーコードの取扱いに関し問題点が生ずることを説明する。

図3は、計算機システム-A105aで取扱われるデータ、プログラムコードと、レジスタ、メモリなどの関連をあらわす図である。

【0026】

計算機システム-A105aは、ハードウェア-A106aと、OS-A107aと、実行環境差異吸収プログラム-A111aと、実行プログラム-A110aで構成されることは前述した。

【0027】

ハードウェア-A106aには、図3に示されるように命令を実行するCPU-A121aと、プログラムやデータを格納するROM-A122aとRAM-A123aが搭載されている。そして、CPU-A121aは、複数の汎用レジスタを有している。

【0028】

すなわち、各用途により、ユーザの記述した引数を渡すための引数格納用途レジスタ124、一時変数を格納する一時変数格納用途レジスタ125、C関数の戻り値を返すための戻り値格納用途レジスタ126、ユーザに提供されるユーザ使用汎用レジスタ127などがある。

【0029】

また、CPU-A121aは、RAM-A123a上に実装されているスタック領域130内の番地を指すスタックポインタレジスタ128、ROM-A122aまたはRAM-A123a上に実装されたプログラムの現在実行中の命令の次のアドレスを指すプログラムカウンタレジスタ129を有している。

【0030】

一般に、C言語などのコンパイラが、CPUの汎用レジスタをデータや引数に割り付けるときには、大別してユーザに対してある関数を呼出した前後の値の存在を保証して、ユーザが使用しても良いとする（すなわち、アセンブラでレジスタの値を直接操作をしても良い）レジスタと、システム上で利用する汎用レジスタがある。この汎用レジスタは、ユーザに対しては、ある関数を呼出した前後の値の存在は保証していない。

【0031】

ここでは、任意のC関数が呼ばれた時点の前後で値の一貫性を保証しないレジスタとして、引数格納用途レジスタ124、一時変数格納用途レジスタ125、戻り値格納用途レジスタ126がある。また、一方、値の一貫性を保証するレジスタは、ユーザ使用汎用レジスタ127、スタックポインタレジスタ128、プログラムカウンタレジスタ129がある。

【0032】

引数格納用途レジスタ124、一時変数格納用途レジスタ125、戻り値格納用途レジスタ126、ユーザ使用汎用レジスタ127、スタックポインタレジスタ128は、通常は、それぞれ、同じ大きさの値を扱える分のレジスタ長になっている。

【0033】

ここで、ROM-A122a上に、実行プログラム-A110aのプログラムコードである実行プログラムコード-A131aと、実行環境差異吸収プログラム-A111aのプログラムコードである実行環境差異吸収プログラムコード-A132aと、OS-A107aのプログラムコードであるOSプログラムコード-A133aが格納されるものとする。これは、図2に示した様に、コンパイラA108aにより、実行環境-A用にコンパイルされたものである。そして、実行プログラム-A110aを実行させるためには、実行プログラムコード-A131aの命令語が解釈・実行される。また、同様に、実行環境差異吸収プログラム-A111aを実行させるためには、実行環境差異吸収プログラムコード-A132a命令語が、OS-A107aのプログラムを実行するためには、O

Sプログラムコード-A133aが、それぞれ解釈・実行される。

【0034】

また、ROM-A122a上は、定数格納領域-A134aが実装されており、OS-A107aの機能に依存したOS依存エラーコード135aと、実行環境差異吸収プログラムで共通化されて提供される共通化エラーコード136が格納されている。

【0035】

さらに、RAM-A123a上には、既述のようにスタック領域130が実装されている。

【0036】

実行プログラムコード-A131aを実行するときには、CPU-A121aにより、実行プログラム開始番地137から始まり、実行プログラム終了番地138までにある命令語が解釈・実行される。その際、プログラムカウンタレジスタ129は、現在動作させている機械語命令の次に動作させる機械語命令の格納番地を指しており、スタックポインタレジスタ128は、現在使用しているスタック領域130内の任意のメモリ番地を指している。

【0037】

プログラムカウンタレジスタ129は、実行プログラムコード-A131a内にプログラムされた共通化インターフェース102部の入口部に到達すると、ジャンプ命令139によって、その共通化インターフェース102に対応する実行環境差異吸収プログラムコード-A132aの先頭番地を指してCPU-A121aを実行する。

【0038】

同じく、プログラムカウンタレジスタ129は、実行環境差異吸収プログラムコード-A132a内にプログラムされたOS-A107aのシステムコールの入口部に到達すると、ソフトウェア割り込み命令またはジャンプ命令140により、そのシステムコールに対応するOSプログラムコード-A133aの先頭番地を指して、CPU-A121aを実行する。

【0039】

OSプログラムコード-A133aが所定の機能を実行し終わると、実行環境差異吸収プログラム-A111aに伝えるため、OS-A107a依存エラーコード135aを、戻り値格納用途レジスタ126に戻り値を格納する。OS-A107a依存エラーコード135aは、OS107aのプログラムが正常終了したのか、OS107aのプログラムや計算機システム-A105a内部でどのような異常が発生したかを示すための情報である。

【0040】

OS-Aプログラムコード-A133aによる戻り値の格納作業を終えると、プログラムカウンタレジスタ129は、実行環境差異吸収プログラムコード-A132aへの戻り先番地142を指して、リターン命令141を実行することにより、実行環境差異吸収プログラムコード-A132aに制御を移す。

【0041】

実行環境差異吸収プログラムコード-A132aに制御が戻ると、戻り値格納用途レジスタ126には、OS-A107a依存エラーコード135aが戻り値として入っている。一方、ROM-A122a内の或るOS-A107a依存エラーコード135aを一時変数格納用途レジスタ125にロードして、そして比較命令144で戻り値格納用途レジスタ126と一時変数格納用途レジスタ125に格納されている値を比較する。そして、全く等価なエラーコードが見つかったときには、実行環境差異吸収プログラムコード-A132aから、実行プログラムコード-A131aに制御が戻るときに、その全く等価なエラーコードに対応する共通化エラーコード136を、戻り値格納用途レジスタ126に格納して、実行プログラム-A110aに渡すことにする。見つからなかったときには、次の或るOS-A107a依存エラーコード135aを一時変数格納用途レジスタ125にロードして、比較命令144を繰り返す。ここで、言う等価なエラーコードを見つけるとは、例えば、OS-A107a依存エラーコード135aが、「メモリエラー」を意味するときには、同じ「メモリエラー」を意味する共通化エラーコードを見つけると言うことである。

【0042】

ここで、ポイントになるのは、実行環境-A104aと実行環境-B104b

とは、エラーコードは、共通化されており、実行プログラム-A110aと実行プログラム-B110bには、同じエラーコードが渡されるということである。そのため、アプリケーションを記述する者にとっては、環境の差異により返されるエラーコードが変わると言うことがなくなり、環境の差異によるコーディングの変更を気にしなくても良いと言うメリットが生ずる。

【0043】

なお、実行開発環境104によつては、OS依存エラーコード135と共通化エラーコード136は、それぞれ、定数格納領域134内ではなく、実行プログラムコード131、実行環境差異吸収プログラムコード132の途中等に実装されることもある。

【0044】

上記従来技術では、実行環境の差異を吸収するレイヤを設けたため、OSに依存しない共通化エラーコードを定義して、ユーザにアプリケーションの移植・記述の容易性を提供することができた。しかしながら、エラーコードのインプリメントに関しては、考慮されていない。

【0045】

というのも、共通化エラーコード136は、ROM-A122aの定数格納領域134、そして、実行プログラムコード131、および、実行環境差異吸収プログラムコード132の内部に格納されるために、ROM効率が悪くなるためである。この問題点は、共通化エラーコード136を大きな格納領域を要する値で定義した場合には、より大きくなるためである。

【0046】

このために、共通化エラーコード136の値を小さくすることが有効であるが、そのようにすると、CPUの命令セットの仕様から新たな問題点を生じる。以下、それについて、図4ないし図7を用いて説明する。

図4は、汎用レジスタでゼロ拡張する際の説明図である。

図5は、ゼロ拡張された汎用レジスタの値を負にする際の説明図である。

図6は、汎用レジスタで符号拡張する際の説明図である。

図7は、符号拡張された汎用レジスタの値をもとに戻す際の説明図である。

【0047】

CPUの命令セットの仕様としては、汎用レジスタに格納可能な範囲より比較的小さな値を格納する際に先ずゼロ拡張するもの、同じく先ず符号拡張するものが現存する。

【0048】

図4のように、命令セットの仕様として、小さな値を汎用レジスタにロードするときには、ゼロ拡張されるものがある。このときには、上の例では、CPU 121 α は、小さな正の値である定数データを、戻り値格納用途レジスタ126または一時変数格納用途レジスタ125の下位部161に格納し、上位部162をゼロ拡張する作業を1命令でおこなうことになる。

【0049】

また、図5に示されように、ゼロ拡張された汎用レジスタの値を負にするためには、上位部に0を2の補数変換処理命令163を施すなどの処理が必要になる。

【0050】

したがって、共通化エラーコード136の値を小さくしても、共通化エラーコード136の値を負のコードで定義して、しかも、レジスタにロードするときの命令セットの仕様が、上記のようにゼロ拡張するものであるときには、2の補数変換処理命令163等が追加することが必要になり、その分だけ、実行プログラムコード-A131aや実行環境差異吸収プログラムコード-A132aの命令数が増え、ROM効率が悪くなると言う問題点が生ずる。

【0051】

一方、図6のように、命令セットの仕様として、小さな値を汎用レジスタにロードするときには、符号拡張されるものがある。このときには、上の例では、CPU 121 β は、小さな正の値である定数データを、戻り値格納用途レジスタ126または一時変数格納用途レジスタ125の下位部161に格納する。そして、小さな値の符号ビットが0（正）のときには、レジスタの上位部を0にし、1（負）のときには、レジスタの上位部を1にする。この一連の動作は、一命令でおこなわれることになる。

【 0 0 5 2 】

図 7 は、負に符号拡張された汎用レジスタの値を、もとに戻す処理を示している。もし、小さな値の定数データの MSB (Most Significant Bit) が 1 であるとする。このときに、この定数データを符号無しデータとみなすか、符号付きデータでみなすかで表している数値の意味が異なってくる。

【 0 0 5 3 】

そして、小さな値の定数データが、MSB が 1 の符号無しデータであり、かつ、図 6 に示したように符号拡張されたときには、上位部の 1 を再び 0 にする必要が生じる。

【 0 0 5 4 】

このように、共通化エラーコード 1 3 6 の値を小さくして、共通化エラーコード 1 3 6 の値を符号無しのコードで定義しても、レジスタにロードするときの命令セットの仕様が、上記のように符号拡張するものであるときには、上位部に 0 をパックする命令 1 6 5 等が追加することが必要になり、その分だけ、実行プログラムコードー A 1 3 1 a や実行環境差異吸収プログラムコードー A 1 3 2 a の命令数が増え、ROM 効率が悪くなるという問題点が生ずる。

【 0 0 5 5 】

本発明は、上記問題点を解決するためになされたもので、その目的は、計算機システムの異なった実行環境で実行させるときのアプリケーション・プログラムにエラーコードを返す際に、既存の命令セットの体系を活かしながら、その共通化コードのインプリメントのための ROM 効率を良くし、むだな命令を挿入することのない計算機システム構成方法を提供することにある。

【 0 0 5 6 】

【課題を解決するための手段】

本発明では、実行環境の差異を吸収するために定義されている共通化エラーコード 1 3 6 を、定数格納領域 1 3 4 に格納しなくても良いようにするため、実行環境差異吸収プログラムコード 1 3 2 からの戻り値と共通化エラーコード 1 3 6 の値を同じ値にし、CPU 1 2 1 の実行できる Immediate ロード命令 1 8 1 に埋め込める範囲内とするという手段を採る。

【0057】

また、命令セットの仕様として、小さな値を汎用レジスタにロードするときには、ゼロ拡張されるようになっているとき、2の補数変換処理命令163等を追加しなくても良いようにするために、共通化エラーコード136を符号無し of 正のデータとして定義する。

【0058】

このように共通化エラーコード136を符号無し of 正のデータと定義することで、2の補数変換処理命令163を追加しなくても良くなる。

【0059】

したがって、実行プログラムコード131や実行環境差異吸収プログラムコード132において、そのような命令を追加しなくても良くなり、ROM効率を良くすることができ、そのような命令を実行しなくて良いため、性能の低下も防止できる。

【0060】

また、命令セットの仕様として、小さな値を汎用レジスタにロードするときには、符号拡張されるようになっているとき、ゼロ埋め込み命令165を追加しなくても良いようにするため、共通化エラーコード136の値を、Immediate設定部182のMSBをゼロとして表せる数値範囲内にする。

【0061】

このように共通化エラーコード136の値を、Immediate設定部182のMSBをゼロとして表せる数値範囲内にすることで、ゼロ埋め込み命令165を追加しなくても良くなる。

【0062】

したがって、共通化エラーコード136を、Immediate設定部182のMSBをゼロとして表せる範囲の値に定義することで、ゼロ埋め込み命令165の追加は考慮せずに済むことが可能となる。

【0063】

したがって、実行プログラムコード131や実行環境差異吸収プログラムコード132において、そのような命令を追加しなくても良くなり、ROM効率を良

くすることができ、そのような命令を実行しなくて良いため、性能の低下も防止できる。

【0064】

上記のように共通化エラーコード136を設定することで、命令セットとして、Immediateロード命令を備えるCPU121に対し、実行プログラムコード131や実行環境差異吸収プログラムコード132に、機械語命令を追加したり、定数格納領域134の容量を増加させることなく、ROM効率を向上させることが可能になる。

【0065】

【発明の実施の形態】

以下、本発明に係る一実施形態を、図8ないし図16を用いて説明する。

〔開発環境とハードウェア〕

先ず、図8および図9を用いて本発明に係る異種実行環境における計算機システム構成方法のソフトウェア開発環境と、それを動作させるためのハードウェアについて説明する。なお、本実施形態では、ソースプログラムの記述を、C言語でおこなった場合を想定するものとする。

図8は、本発明に係る異種実行環境における計算機システム構成方法のソフトウェア開発環境の処理の流れを説明するための図である。

図9は、本発明に係る異種実行環境における計算機システム構成方法を実行するためのハードウェア構成図である。

【0066】

本発明は、複数の異なったOSでの開発環境210を前提としている。

【0067】

ソフトウェアの開発者は、アプリケーションのプログラムが記述したアプリケーション本体100と、共通化関数102のソースコードを入力して、通常の開発手順の通り、プリプロセッサ211、Cコンパイラ212、アセンブラ213、リンカ214と言うソフトウェアツールにより、実行プログラムを作成する。

【0068】

プリプロセッサ211は、条件によって加工したり、読み飛ばしたり、他プロ

グラムを組み込んだりマクロ定義処理などをおこなうCコンパイラに入る前の前処理のためのツールである。Cコンパイラ212は、記述されたテキストコードに対して、構文解析・意味解析などをおこない、適当なアセンブラコードを生成するものである。アセンブラ213は、アセンブラコードをCPU121で実行させる機械語に変換するツールである。

【0069】

リンカ214は、ライブラリ190から適当なプログラムコードを選び、数種のプログラムコードファイルを結合して最終的にハードウェア上で実行させる実行プログラムを作成する。本実施形態の例では、アプリケーション本体100をコンパイルした実行プログラムコード131、実行環境差異吸収プログラムコード132、および、OSのコードといった必要なモジュールと、定数格納領域134、スタック領域130などのデータ部分とを結合させて、実際にCPU121に実行させる実行形式のモジュールを作成する。このライブラリ190には、実効環境に依存する実行環境プログラムコードが含まれていて、機能に応じてリンカ214により適当なプログラムが取りこまれて結合される。

【0070】

OSコンフィグレータ191は、アプリケーションソフトウェア開発者に、ターゲットとするOS107の諸機能を選択および設定させカスタマイズ化されたOS107のプログラムコードを作成するソフトウェアツールである。

【0071】

ハードウェア106は、CPU121、Device201、ワーキングメモリ202から構成されている。

【0072】

最終的にCPU121上で実行する実行プログラムとOS107は、ワーキングメモリ202に配置される。

【0073】

ワーキングメモリ202は、ROMとRAMであり、実行形式のモジュールとして、ユーザの実行プログラムと実行環境差異吸収プログラム、OSのモジュールが格納される。また、データとしては、定数格納領域134、スタック領域1

30が格納される。

【0074】

本実施形態のソフトウェア開発環境は、メインフレーム、パーソナルコンピュータやワークステーション等の汎用計算機上に実装することが可能である。

【0075】

最終的に実行プログラムを動作させることのできるハードウェア106を作成する手順を示すと以下の様になる。

【0076】

すなわち、アプリケーションソフトウェア開発者は、汎用計算機上に定義されているターゲットのOS107に対応するOS別プログラム開発環境210を選択する。そして、選択されたOS別プログラム開発環境210に付随しているOSコンフィグレータ191を用いてカスタマイズしたOS107のプログラムコードを作成する。

【0077】

次に、そのOS107のOSプログラムコード133と実行プログラムコード131と実行環境差異吸収プログラムコード132をリンクしてできあがったプログラムコード全体をワーキングメモリ202に組み込む。また、予めOS107のOSプログラムコード133が搭載されたワーキングメモリ202に、後から実行プログラムコード131と実行環境差異プログラムコード132をリンクしてできあがったプログラムコードを組み込み、動的にリンクしながら実行させるようにしても良い。

【0078】

ここで、ハードウェア106は、実行プログラムを実行できるような機構を備えていれば良いが、本実施形態では、主にプログラムをワーキングメモリに組み込んだシステムLSIなどを想定している。

【0079】

以下、図9を用いてこのハードウェア106の構成の例について詳細に説明する。

【0080】

CPU 1 2 1 は、命令を実行し、計算をおこなうハードウェア 1 0 6 のメインのコンポーネントであり、図 3 で説明したような汎用レジスタ群 2 2 2 を備えている。汎用レジスタ群 2 2 2 に含まれるのは、引数格納用途レジスタ 1 2 4、一時変数格納用途レジスタ 1 2 5 と、戻り値格納用途レジスタ 1 2 6、ユーザ使用汎用レジスタ 1 2 7、スタックポインタレジスタ 1 2 8、プログラムカウンタレジスタ 1 2 9 などである。

【 0 0 8 1 】

RAM 1 2 3 は、いつでも書き込みが可能なメモリエリアである。この RAM 1 2 3 に、スタック領域 2 0 5 が実装されている。ROM 1 2 2 は、通常は書きこむことができず、読みだし専用のメモリエリアである。例えば、LSI の製造時にのみ情報を書きこみ、通常の実行時には、このメモリエリアからは、情報を読むだけである。ROM 1 2 2 には、定数格納領域 1 3 4 が実装される。そして、ROM 2 0 3 b または RAM 1 2 3 の一部には、実行プログラム 1 1 0 と実行環境差異プログラム 1 1 1 と OS のプログラムコードが組み込まれ、CPU 1 2 1 に読み出されて実行される。

【 0 0 8 2 】

デバイス 2 0 1 は、各種機能を持つハードウェア 1 0 6 のコンポーネントであり、ハードウェア 1 0 6 がシステム LSI の場合には、LSI 上に構成される特殊機能の処理回路である。具体的には、I/O (Input/Output) , ASIC (Application Specific Integrated Circuits) , FPGA (Field Programmable Gate Arrays) , DSP (Digital Signal Processor) などが考えられる。

【 0 0 8 3 】

I/O は、例えば、A/D 変換器、D/A 変換器、RS 2 3 2 - C 処理回路、SCSI 処理回路である。ASIC は、例えば、MPEG Video 符号器、MP 3 復号器等の専用処理回路である。FPGA は、ハードウェア構成を可変にできる IC を言う。DSP は、ディジタル信号処理専用の IC である。

【 0 0 8 4 】

これらのコンポーネントは、信号の共通の通り道であるバス 2 0 3 により、情報のやり取りをする。

〔Immediateロード命令〕

本発明は、実行環境の差異を吸収するレイヤにおいて、共通エラーコードを返すときに、Immediateロード命令を用いることに特徴がある。そのため、次に、図10ないし図12を用いてImmediateロード命令の形式と機能について説明する。

図10は、Immediateロード命令の形式を示した図である。

図11は、Immediateロード命令の他の形式を示した図である。

図12は、本発明に係る計算機システム構成方法において、共通化エラーコードとImmediateロード命令の関係を示す図である。

【0085】

Immediateロード命令181は、Immediate値（即値、定数データ）を、指定されたレジスタに持ってくる命令である。その命令語としての形式は、例えば、図10に示されるようにImmediate設定部182と汎用レジスタ選択部183とオペコード部184で構成される。オペコード部184は、命令の種類および機能を表すコードが入る部分である。汎用レジスタ選択部183は、ロードするレジスタの番号を指定する。Immediate設定部182は、Immediate値を保持する部分である。

【0086】

CPU121は、オペコード部184から、命令の種類および機能がImmediateロード命令181であることを判別し、Immediate設定部182に指定されたImmediate値を、汎用レジスタ選択部183で指定された汎用レジスタに格納する。

【0087】

したがって、コンパイルする以前に共通化エラーコード136の値と実行環境差異吸収プログラムコード132の戻り値をImmediate設定部182の扱える範囲の数値と定義しておく、例えば、Cコンパイラ108は、定数格納領域データロード命令143の代わりに、実行環境の差異を吸収するための共通化エラーコード136を、ユーザが記述したアプリケーションに返すために、戻り値格納レジスタに設定する命令などについて、その共通化エラーコード136

を Immediate 設定部 182 に保持する Immediate ロード命令 181 を作成してくれる。

【0088】

本発明では、図 12 に示したように、Immediate ロード命令 181 の Immediate 設定部 182 に共通化エラーコード 136 を定義しておき、その Immediate 設定部 182 の MSB を 0 として表せる正の値の範囲に設定する。

【0089】

また、CPU の命令セットとして、図 10 に示した形式の Immediate ロード命令 181 i の他に、図 11 に示した形式を持ち、ロードする汎用レジスタを複数設定できる Immediate ロード命令 181 j を有することもあるが、この Immediate ロード命令 181 j を、共通化エラーコード 136 の設定のために、Immediate ロード命令 181 i と全く同様に使用することができる。

〔共通化エラーコードの具体例〕

次に、図 13 を用いて本発明で設定する共通化エラーコードの具体例について説明する。

図 13 は、本発明の計算機システム構成方法に係る共通化エラーコードの具体例を示す図である。

【0090】

図 13 に示した共通化エラーコードは、Immediate ロード命令 181 の Immediate 選択部 182 として、例えば、1 バイト長、2 バイト長の場合が想定できる。

〔プログラムコードと Immediate ロード命令の関係〕

次に、図 14 を用いてプログラムコードと Immediate ロード命令の関係を説明する。

図 14 は、プログラムコードに Immediate ロード命令が埋め込まれているときの計算機システム-A105a で取扱われるデータ、プログラムコードと、レジスタ、メモリなどの関連をあらわす図である。

【0091】

実行プログラムコード131は、アプリケーション本体をコンパイルして作られる実行形式のモジュールであり、実行環境差異吸収プログラムコード132は、共通関数をコンパイルして、必要なモジュールをリンクして得られる実行形式のモジュールである。これらの実行形式のモジュールは、ハードウェア106のROM122またはRAM123上に実装される。

【0092】

図14に示された実行プログラムコード131および実行環境差異吸収プログラムコード132には、図4により説明した定数格納領域データロード命令143の代わりに、Immediateロード命令181が利用される。そのImmediateロード命令181内部のImmediate選択部182は、MSB164が0になっている。そして、MSB164以外のImmediate部281には、共通化インターフェース102により定められている共通化エラーコード136が埋め込まれている。

【0093】

したがって、この場合に、共通化エラーコードを定数格納領域に保持する必要がなくなる。

〔Cソースプログラムの例〕

次に、図15および図16を用いて本発明の計算機システム構成方法の開発環境で準備されるCソースプログラムの例について説明する。

図15は、本発明の開発環境の共通化関数のCソースプログラムの例を示した図である。

図16は、本発明の開発環境のアプリケーションソフトウェアのCソースプログラムの例を示した図である。

【0094】

共通化関数101は、ユーザにシステムの機能を提供する関数であり、ユーザに対する関数インタフェースとしては、共通化インターフェース102を公開する。共通化インターフェース102は、実質上、C言語仕様でいう関数定義のことであり、インターフェース機能名称242、引数のデータ型243、戻り値の

データ型 244 を定義している。この図 15 に示す例では、インターフェース機能名称 242 は、「WPR_Wait_Mail」であり、引数は二つであり、引数のデータ型 243 として、共に「int」型、戻り値のデータ型 244 は、「unsigned int」型である。

【0095】

共通化関数の C ソースプログラム 241 には、実行環境の差異を吸収するためのコーディングがなされる。例えば、計算機システム-A105a の実行開発環境-A104a 上で動作させる場合には、その OS-A107a のシステムコール 245a をコールするコーディングがなされる。また、計算機システム-B105b の実行開発環境-B104b 上で動作させる場合には、その OS-B107b のシステムコール 245a をコールするコーディングがなされる。

【0096】

このように共通化関数 101 を用いることにより、ユーザに対して実行環境の差異を隠蔽することが可能であり、ユーザのアプリケーションからすると共通化インターフェース 102 とその関数の機能のみを意識すれば良い。

【0097】

実行開発環境-A104a 上で動作させる共通化関数 101 の内部の処理としては、図 15 に示されるように、例えばシステムコール 245a を介して所望の処理をおこない、その結果、OS-A107a から戻されるシステムコールの OS-A107a 依存エラーコード 135u, 135v を判別して、共通化インターフェース 102 定義の共通化エラーコード 136u, 136v, 136w を共通化関数 101 から戻り値として返すことにする。これまで述べてきたように、共通化エラーコード 136 の数値は、実行開発環境 104 および計算機システム 105 に全く依存しないように、共通化した定義を用いる。

【0098】

そして、共通化エラーコード 136 の値を、Immediate ロード命令 181 における Immediate 設定部 182 の MSB を 0 として表せる数値範囲であり、かつ、符号無しと定義するので、戻り値のデータ型 244 と共に少なくとも C 言語仕様でデータ型として用いられる unsigned 型で宣言する。図 15 の

共通化関数 1 0 1 では、汎用レジスタ群 2 2 2 と同じ b i t 長とし、unsigned i n t 型で記述しているが、共通化エラーコード 1 3 6 を unsigned short 型、 unsigned char 型といった汎用レジスタ群 2 2 2 の b i t 長よりも短く定義しても構わない。

【 0 0 9 9 】

次に、この共通化関数 1 0 1 を呼出すアプリケーションソフトウェアの C ソースプログラムを示すと、図 1 6 のようになる。この例では、アプリケーションプログラム関数 2 6 1 「ApplicationTask」が、共通化関数 1 0 1 「WPR_Wait_Mail」を呼出している。

【 0 1 0 0 】

このアプリケーションプログラム関数 2 6 1 は、内部で共通化関数 1 0 1 を呼出すだけであり、実行環境に依存するコーディング入れないため、アプリケーションレベルの開発においては、全く実行環境、OS、実行される計算機を意識する必要はない。

【 0 1 0 1 】

図 1 6 に示した例では、アプリケーションプログラム関数 2 6 1 「ApplicationTask」が、共通化関数 1 0 1 「WPR_Wait_Mail」を呼出し、戻り値を error_state という変数 2 6 3 に入れて、その後、それが共通化エラーコード 1 3 6 u, 1 3 6 v のどれにあたるを判定し、その処理を分岐させている。戻り値を error_state という変数の型は、当然、共通化関数 1 0 1 「WPR_Wait_Mail」の戻り値の型と同じであり、共通化エラーコード 1 3 6 の定義した型である unsigned int 型である。また、共通化エラーコード 1 3 6 の型が、他の unsigned short 型、unsigned char 型等のときには、その宣言した変数 2 6 3 に格納する。変数 2 6 3 は、アプリケーションプログラム関数 2 6 1 で C 言語仕様のローカル変数宣言またはグローバル変数宣言されているかどうかに従い、OS 別プログラム開発環境 2 0 1 によりそれぞれ、局所変数及び一時変数としてスタック領域 1 3 0 またはグローバル変数としてスタック領域 1 3 0 以外の RAM 領域に実装される。

【 0 1 0 2 】

【発明の効果】

本発明によれば、計算機システムの異なった実行環境で実行させるときのアプリケーション・プログラムにエラーコードを返す際に、既存の命令セットの体系を活かしながら、その共通化コードのインプリメントのためのROM効率を良くし、むだな命令を挿入することのない計算機システム構成方法を提供することができる。

【図面の簡単な説明】

【図 1】

特開平 8 - 6 3 3 6 3 号公報の「仮想実行環境システム」のシステム概要図である。

【図 2】

異種実行環境においてソフトウェア開発をおこなうときの概念図である。

【図 3】

計算機システム-A 1 0 5 a で取扱われるデータ、プログラムコードと、レジスタ、メモリなどの関連をあらわす図である。

【図 4】

汎用レジスタでゼロ拡張する際の説明図である。

【図 5】

ゼロ拡張された汎用レジスタの値を負にする際の説明図である。

【図 6】

汎用レジスタで符号拡張する際の説明図である。

【図 7】

符号拡張された汎用レジスタの値をもとに戻す際の説明図である。

【図 8】

本発明に係る異種実行環境における計算機システム構成方法のソフトウェア開発環境の処理の流れを説明するための図である。

【図 9】

本発明に係る異種実行環境における計算機システム構成方法を実行するためのハードウェア構成図である。

【図 1 0】

Immediateロード命令の形式を示した図である。

【図 1 1】

Immediateロード命令の他の形式を示した図である。

【図 1 2】

本発明に係る計算機システム構成方法において、共通化エラーコードとImmediateロード命令の関係を示す図である。

【図 1 3】

本発明の計算機システム構成方法に係る共通化エラーコードの具体例を示す図である。

【図 1 4】

プログラムコードにImmediateロード命令が埋め込まれているときの計算機システム-A105aで取扱われるデータ、プログラムコードと、レジスタ、メモリなどの関連をあらわす図である。

【図 1 5】

本発明の開発環境の共通化関数のCソースプログラムの例を示した図である。

【図 1 6】

本発明の開発環境のアプリケーションソフトウェアのCソースプログラムの例を示した図である。

【符号の説明】

- 1 0 … アプリケーションソフトウェア
- 1 1 … プログラム本体
- 1 2 … 置換情報記述部
- 2 0 … 仮想実行環境実現システム
- 2 1 … 翻訳部（既存OS用コンパイラ）
- 2 2 … 実行部（仮想実行環境実現部）
- 3 0 … 実行可能プログラム
- 6 0 … 既存オペレーティング・システム
- 1 0 0 … アプリケーション本体
- 1 0 1 … 共通化インターフェース

- 1 0 2 … 共通化関数
- 1 0 3 … 実行環境プログラム
- 1 0 4 … 実行環境
- 1 0 5 … 計算機システム
- 1 0 6 … ハードウェア
- 1 0 7 … OS
- 1 0 8 … Cコンパイラ
- 1 0 9 … リンカ
- 1 1 0 … 実行プログラム
- 1 1 1 … 実行環境差異吸収プログラム
- 1 2 1 … CPU
- 1 2 2 … ROM
- 1 2 3 … RAM
- 1 2 4 … 引数格納用途レジスタ
- 1 2 5 … 一時変数格納用途レジスタ
- 1 2 6 … 戻り値格納用途レジスタ
- 1 2 7 … ユーザ使用汎用レジスタ
- 1 2 8 … スタックポインタレジスタ
- 1 2 9 … プログラムカウンタレジスタ
- 1 3 0 … スタック領域
- 1 3 1 … 実行プログラムコード
- 1 3 2 … 実行環境差異吸収プログラムコード
- 1 3 3 … OSプログラムコード
- 1 3 4 … 定数格納領域
- 1 3 5 … OS依存エラーコード
- 1 3 6 … 共通化エラーコード
- 1 3 7 … 実行プログラム開始番地
- 1 3 8 … 実行プログラム終了番地
- 1 3 9 … ジャンプ命令

- 1 4 0 … ソフトウェア割り込み命令またはジャンプ命令
- 1 4 1 … リターン命令
- 1 4 2 … 実行環境差異吸収プログラムコードへの戻り先番地
- 1 4 3 … 定数格納領域データロード命令
- 1 4 4 … 比較命令
- 1 4 5 … 実行プログラムへの戻り先番地
- 1 6 1 … 下位部
- 1 6 2 … 上位部
- 1 6 3 … 2 の補数変換処理命令
- 1 6 4 … M S B
- 1 6 5 … ゼロ埋め込み命令
- 1 8 1 … I m m e d i a t e ロード命令
- 1 8 2 … I m m e d i a t e 設定部
- 1 8 3 … 汎用レジスタ選択部
- 1 8 4 … オペコード部
- 1 9 0 … ライブラリ
- 1 9 1 … O S コンフィグレータ
- 2 0 1 … デバイス
- 2 0 2 … ワーキングメモリ
- 2 0 3 … バス
- 2 1 0 … O S 別プログラム開発環境
- 2 1 1 … プリプロセッサ
- 2 1 2 … C コンパイラ
- 2 1 3 … アセンブラ
- 2 1 4 … リンカ
- 2 2 2 … 汎用レジスタ群
- 2 4 1 … 共通化関数の C ソースプログラム
- 2 4 2 … インターフェース機能名称
- 2 4 3 … 引数のデータ型

2 4 4 … 戻り値のデータ型

2 4 5 … システムコール

2 6 1 … アプリケーションプログラム関数

2 6 2 … 引数

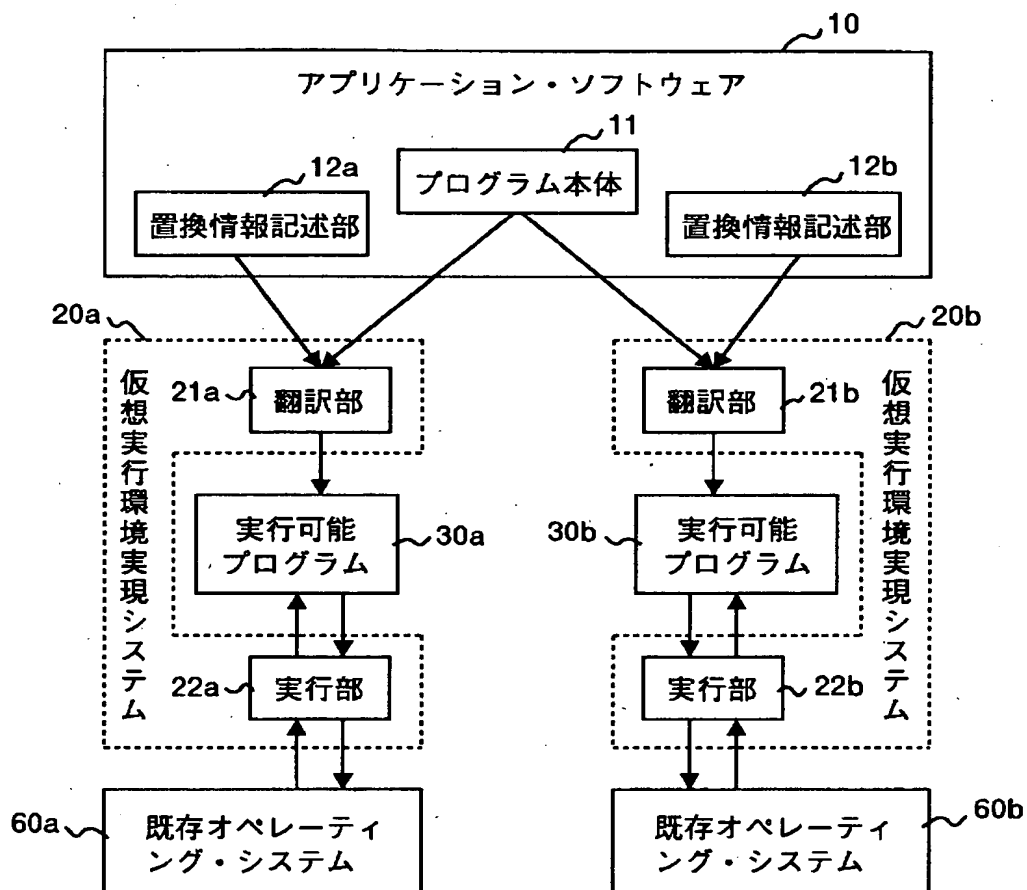
2 6 3 … 変数

2 8 1 … MSB 以外の Immediate 部

【書類名】 図面

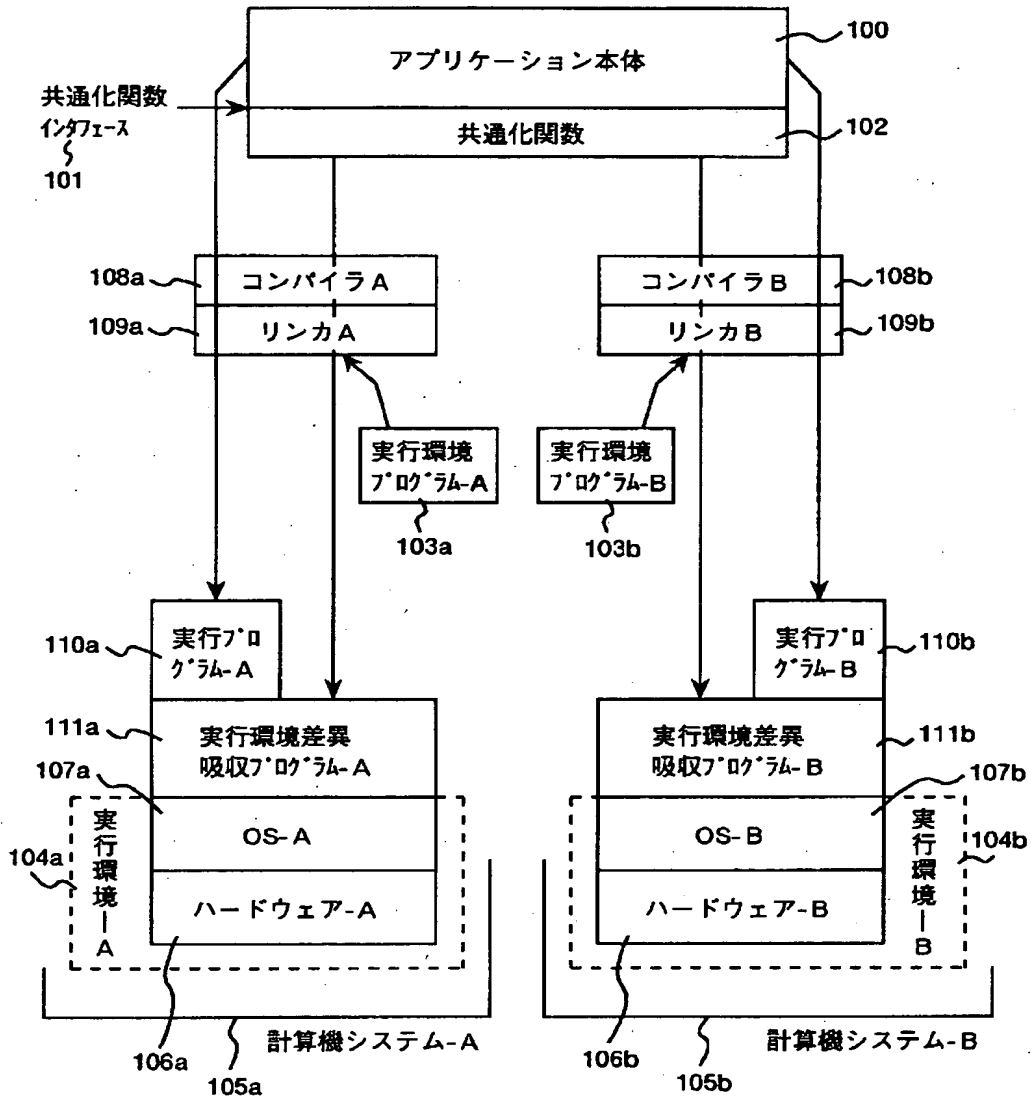
【図 1】

図 1



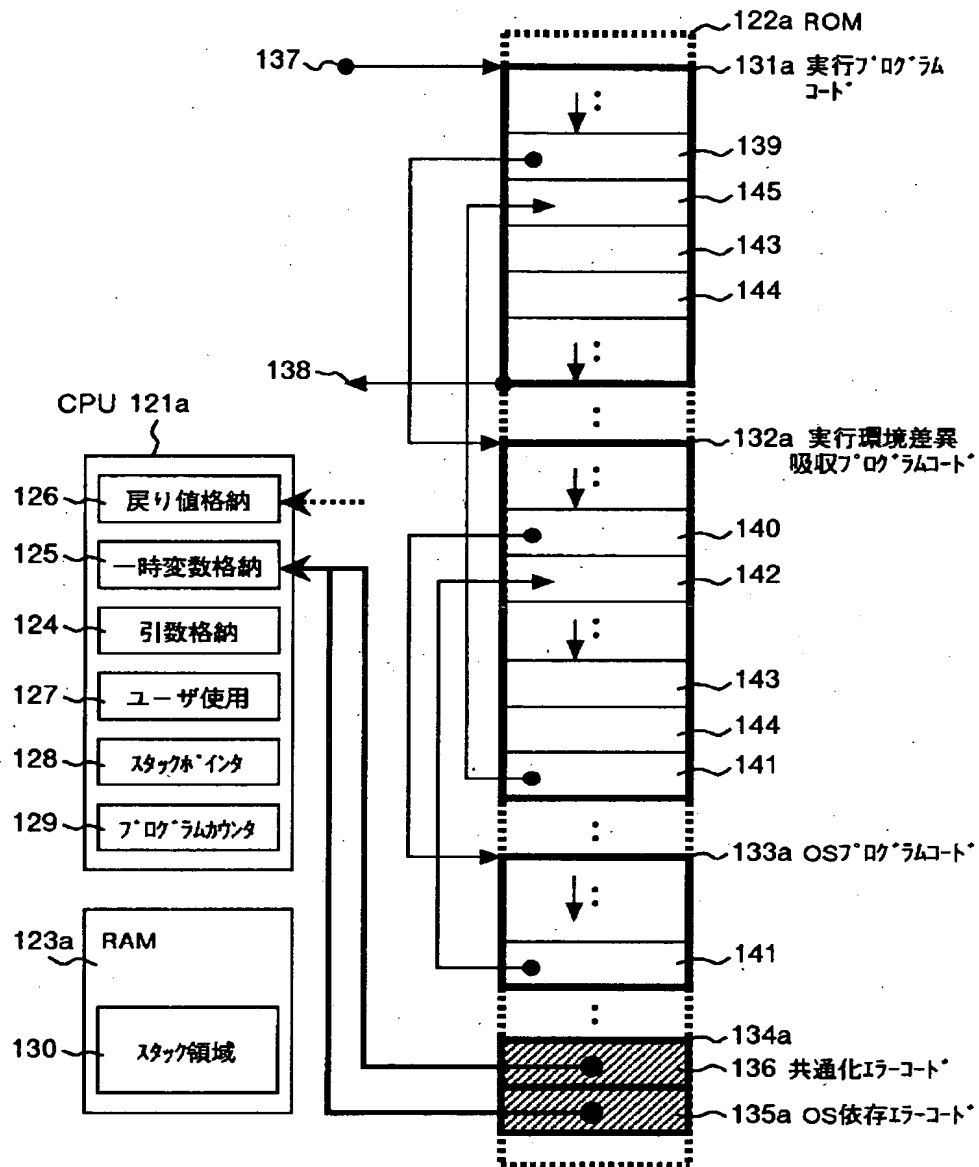
【図 2】

図 2



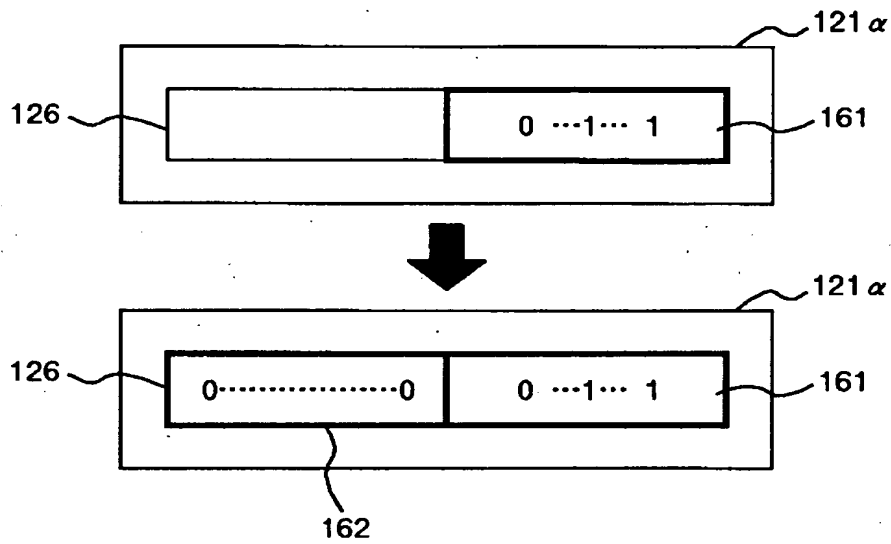
【図 3】

図 3



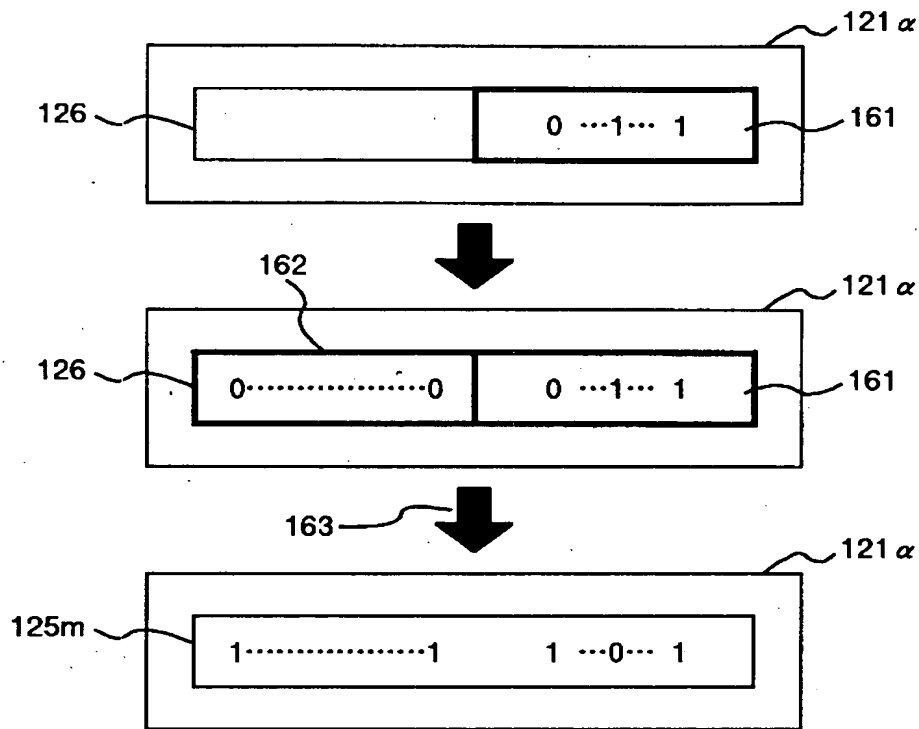
【図 4】

図 4



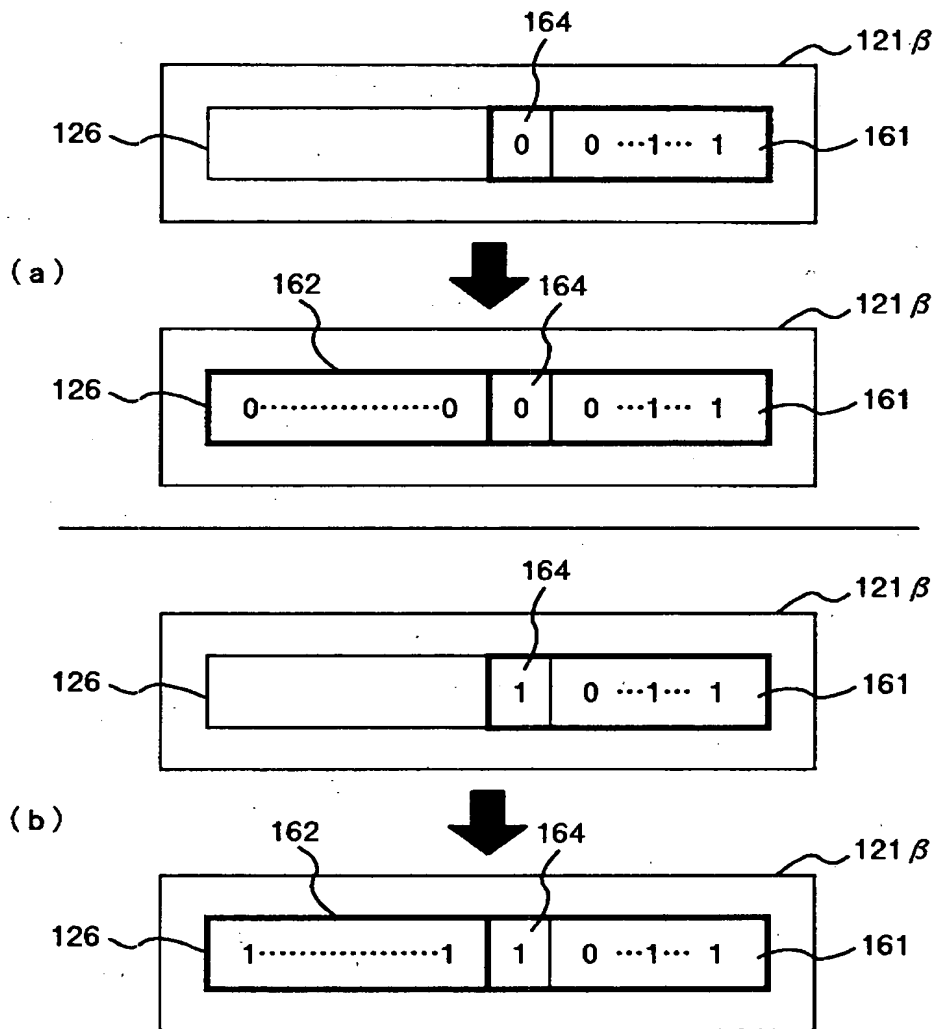
【図 5】

図 5



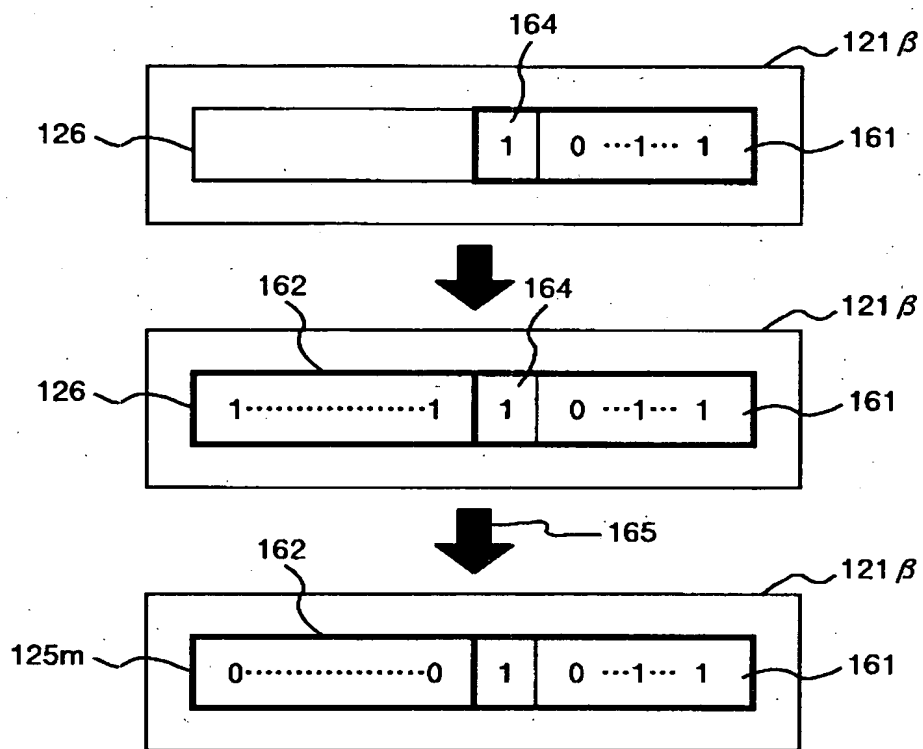
【図 6】

図 6



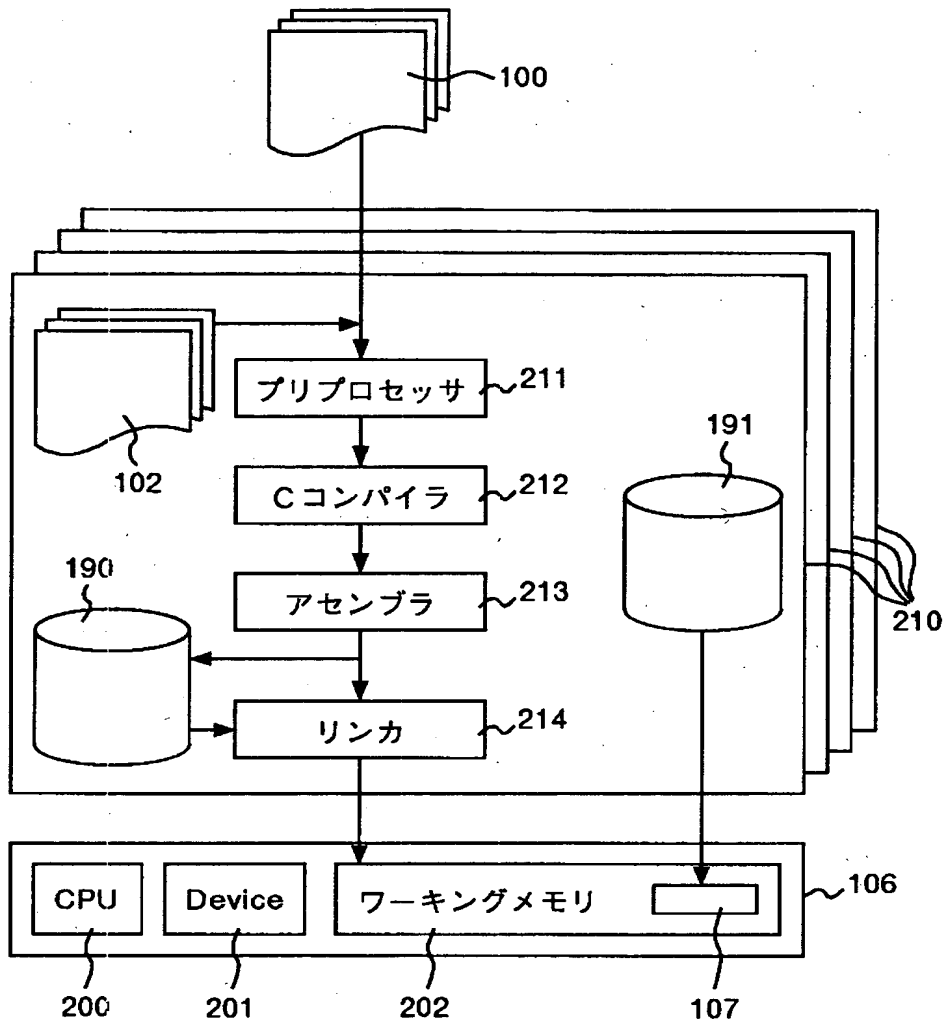
【図 7】

図 7



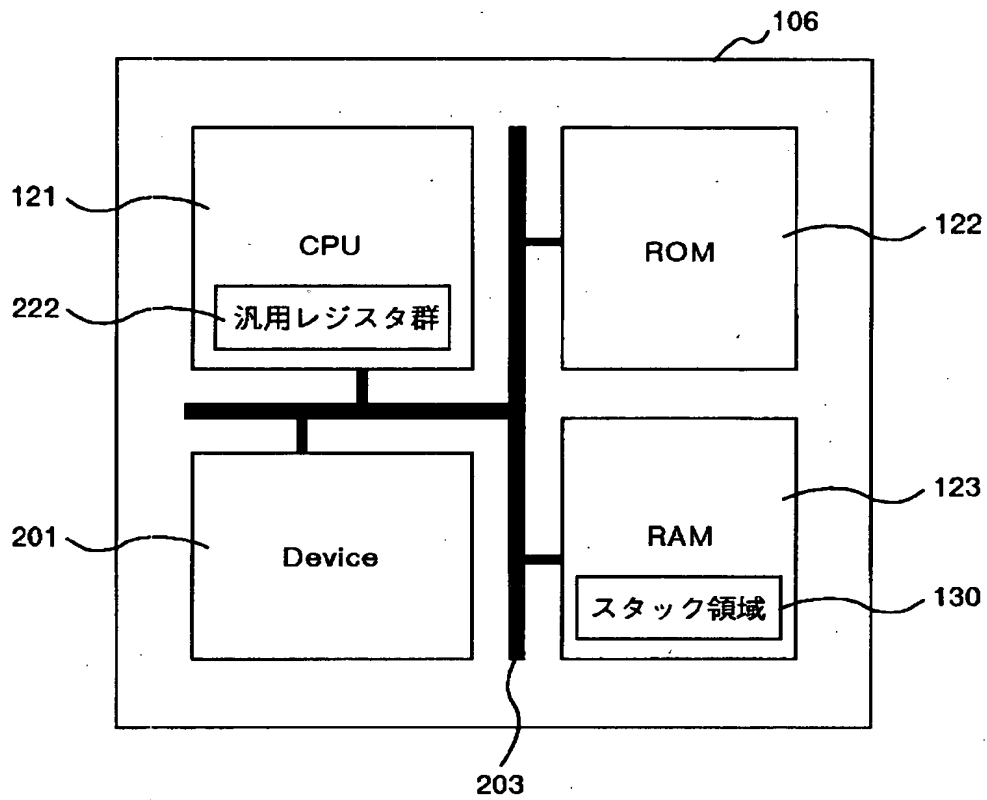
【図 8】

図 8



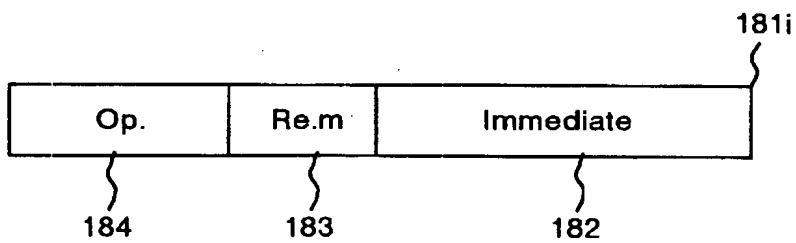
【図 9】

図 9



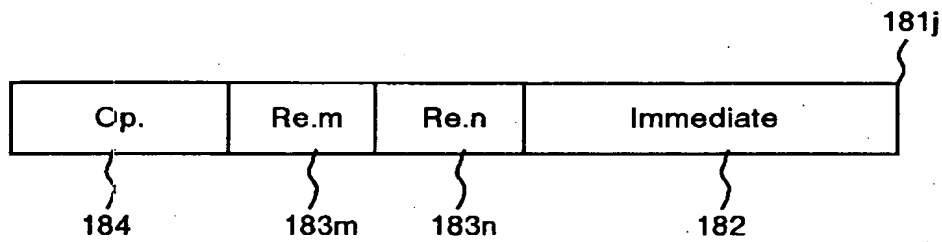
【図 10】

図 10



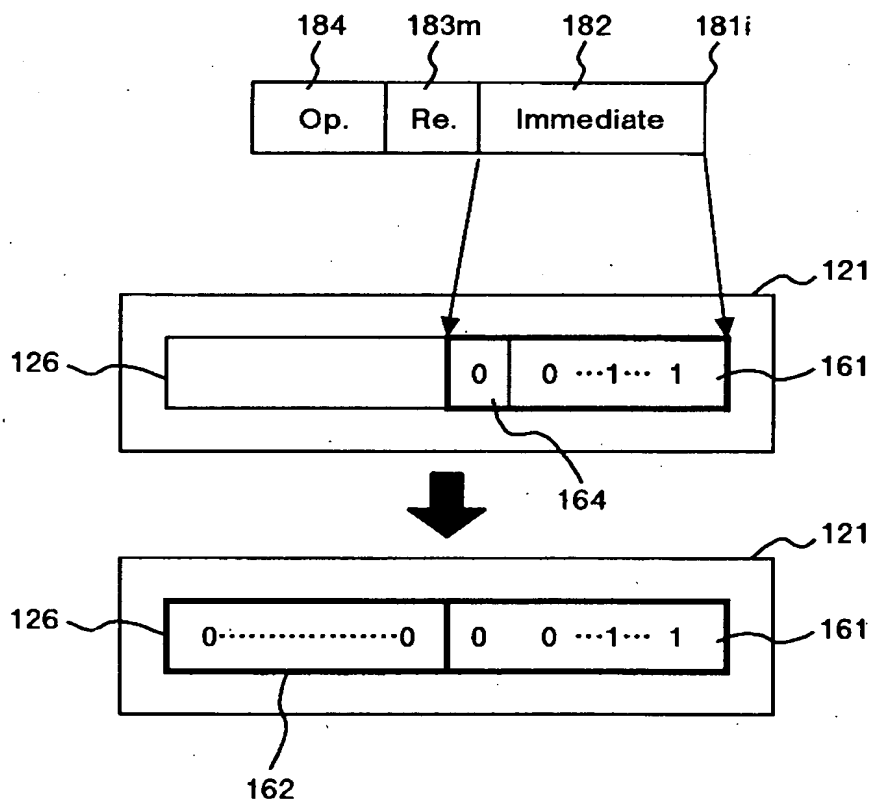
【図 1 1】

図 1 1



【図 1 2】

図 1 2



【図 1 3】

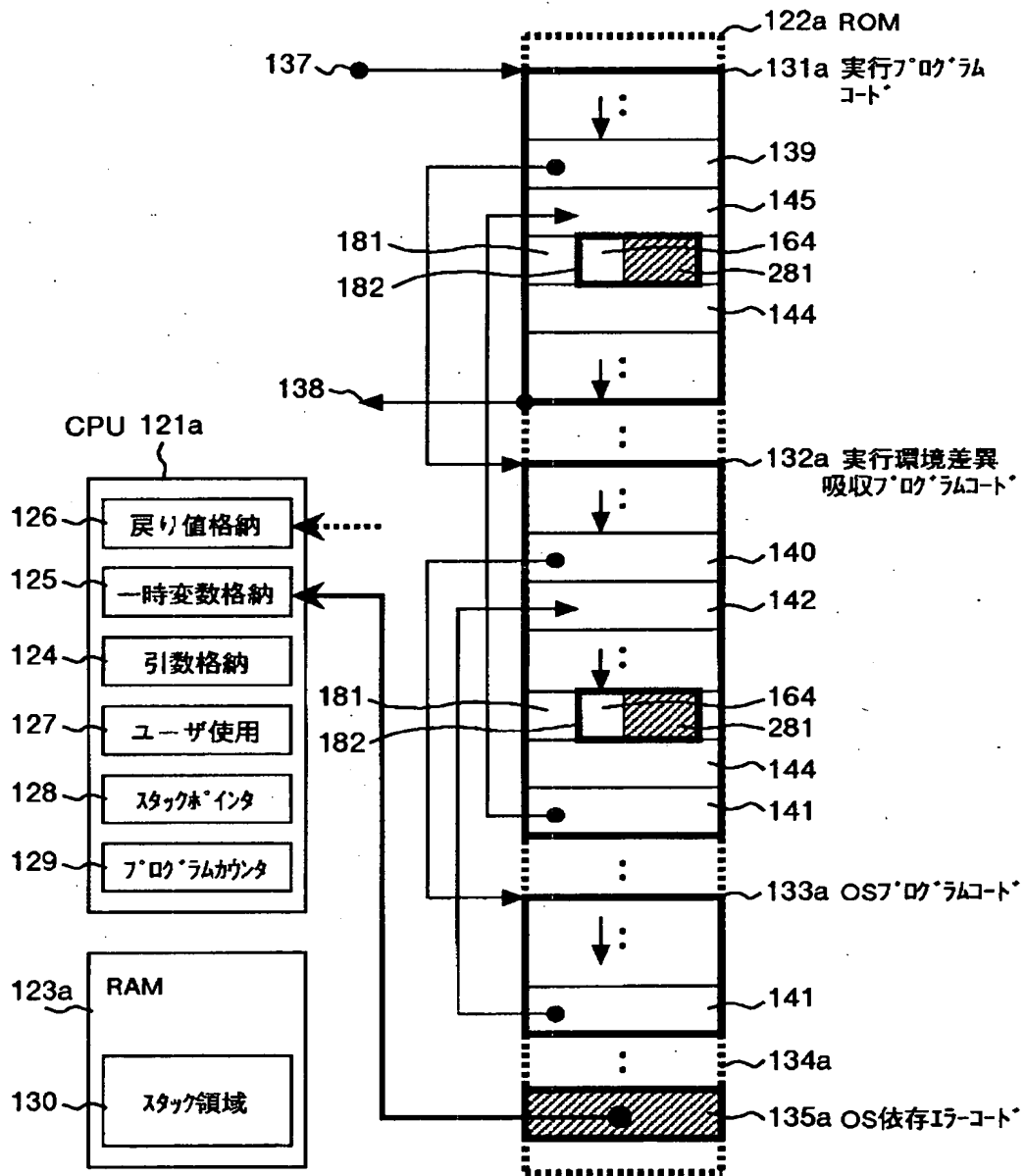
図 1 3

エラーコード名称	値	意味
WRP_OK	0X 00	正常
WRP_ER_STACK	0X 01	スタックエラー
WRP_CREATE	0X 03	タスク生成失敗
WRP_ER_MEM	0X 05	メモリエラー
WRP_ER_OTHER	0X 06	その他エラー
WRP_ER_DELETE	0X 0D	タスクが削除されているので、 実行不可

0X：16進数を表す

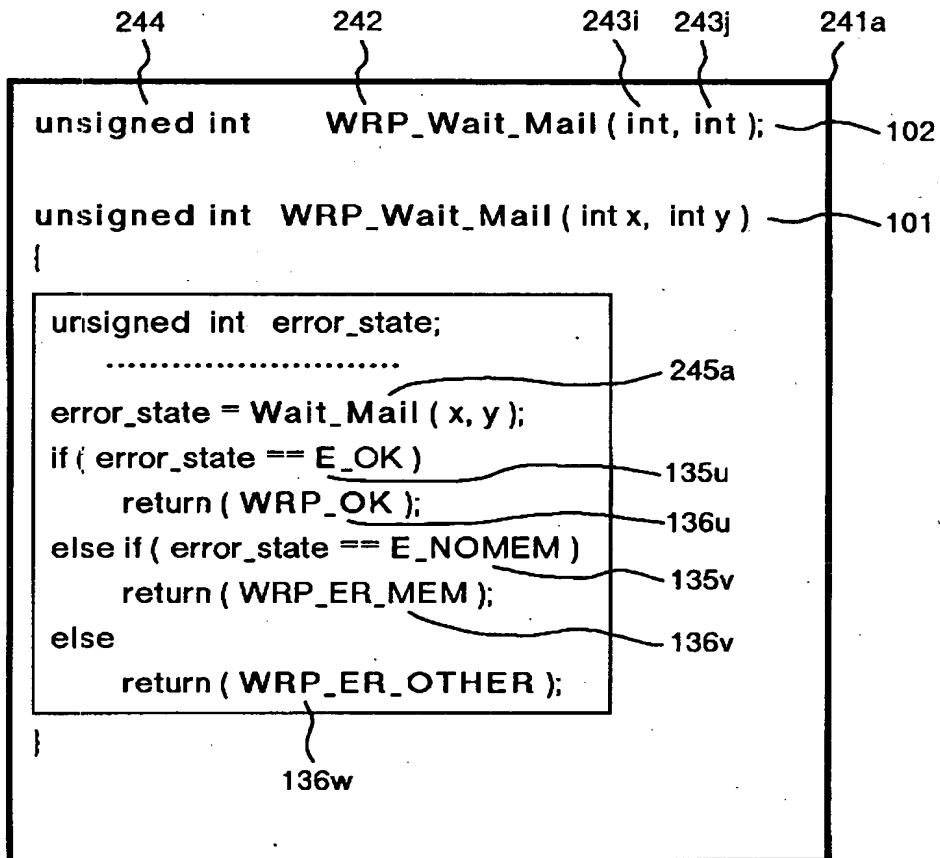
【図 14】

図 14



【図 15】

図 15



【図 16】

図 16

```

    244      242      243i  243j  11
    {
    unsigned int  WRP_Wait_Mail ( int, int ); 102

    void ApplicationTask ( void ) 261
    {
        int          m,n;
        unsigned int  error_state; 263

        .....
        .....
    101 error_state = WRP_Wait_Mail ( m, n ); 262p
        if ( error_state == WRP_OK ) 262q
            ..... 136u
        else if ( error_state == WRP_ER_MEM ) 136v
            .....
        else
            .....
        .....
        .....
    }
    
```

【書類名】 要約書

【要約】

【課題】 計算機システムの異なった実行環境で実行させるときのアプリケーション・プログラムにエラーコードを返す際に、既存の命令セットの体系を活かしながら、その共通化コードのインプリメントのためのROM効率を良くし、むだな命令を挿入することのないようにする。

【解決手段】 実行環境差異吸収プログラムから、アプリケーション・プログラムに対応する実行プログラムに返すための共通化エラーコードを、CPUの命令セットと定義されているImmediateロード命令で設定可能な数値範囲内の値として定めて、共通化エラーコードを、Immediateロード命令の命令コード内に保持するようにする。特に、共通化エラーコードの値をImmediateロード命令の設定部のMSBを0となる範囲内に収まるようにする。

【選択図】 図 1 2

出 願 人 履 歴 情 報

識別番号 [000005108]

1. 変更年月日	1990年 8月31日
[変更理由]	新規登録
住 所	東京都千代田区神田駿河台4丁目6番地
氏 名	株式会社日立製作所